Relating queries and documents

• In the last lecture, we saw how webpages may be ranked according to how important they appear.
  • This was done by taking advantage of the structure of the web (i.e., links in and links out).

• What if we don’t have very much structure?

• How do we relate queries and documents in the first place?
The vector space model

- In the vector space model, queries and documents are both represented by unit-length vectors in word space.
  - Each dimension is a word in the document collection.
  - The domain of each dimension can be, e.g.,
    - $0/1$: absent/present
    - $\mathbb{N}$: term frequency of word $i$ in document $j$ ($tf_{ij}$).
    - $\mathbb{R}_0^+$: damped weight, e.g.,
      \[
      = \begin{cases} 
      1 + \log tf_{ij} & \text{if } tf_{ij} > 0 \\
      0 & \text{if } tf_{ij} = 0 
      \end{cases}
      \]

  Note: vectors in the above domains can easily be converted to unit vectors if desired.
  Note 2: you don’t normally need these vectors to be unit length – sometimes you explicitly don’t.
The vector space model

• Once vectors are determined for the query and the available documents, we can determine similarity according to the cosine method.
  • Vectors that are near each other (within a certain angular radius from the query) are considered relevant.
The cosine measure

• The cosine measure (a.k.a., ‘normalized correlation coefficient’) is

\[ \cos(\mathbf{q}, \mathbf{d}) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2} \sqrt{\sum_{i=1}^{n} d_i^2}} \]

where \(\mathbf{q}\) and \(\mathbf{d}\) are \(n\)-dimensional vectors for the query and document, respectively.
The cosine measure

• The cosine measure
  (a.k.a., ‘normalized correlation coefficient’) is
  \[
  \cos(\vec{q}, \vec{d}) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2} \sqrt{\sum_{i=1}^{n} d_i^2}}
  \]
  where \(\vec{q}\) and \(\vec{d}\) are \(n\)-dimensional vectors for the query and document, respectively.

• Larger values of \(\cos(\vec{q}, \vec{d})\) means stronger correlation, so \(\vec{q}\) is ‘closer’ to \(\vec{d}_1\) than \(\vec{d}_2\) iff \(\cos(\vec{q}, \vec{d}_1) > \cos(\vec{q}, \vec{d}_2)\).
Term weighting

• What if we want to **weight** words in the vector space model?

  • **Term frequency,** $tf_{ij}$: number of occurrences of word $w_i$ in document $d_j$.

  • **Document frequency,** $df_i$: number of documents in which $w_i$ appears.

  • **Collection frequency,** $cf_i$: total occurrences of $w_i$ in the collection.
Term frequency

• **Higher** values of $tf_{ij}$ (for contentful words) suggest that word $w_i$ is a good indicator of the content of document $d_j$.
  • When comparing the relevance of a document $d_j$ to a keyword $w_i$, $tf_{ij}$ should be **maximized**.

• We often **dampen** $tf_{ij}$ to temper these comparisons.
  • E.g., even if $tf_{ij} = 3tf_{ik}$, *empirically* we don’t want to say that document $d_j$ is *thrice* as relevant as document $d_k$.
    • $tf_{dampen} = 1 + \log(tf)$, if $tf > 0$. 
Document frequency

- The **document frequency**, $d_{f_i}$, is the number of documents in which $w_i$ appears.
  - **Meaningful** words may occur repeatedly in a related document, but **functional** (or less meaningful) words may be **distributed** evenly over all documents.

<table>
<thead>
<tr>
<th>Word</th>
<th>Collection frequency</th>
<th>Document frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel</td>
<td>10,440</td>
<td>3997</td>
</tr>
<tr>
<td>try</td>
<td>10,422</td>
<td>8760</td>
</tr>
</tbody>
</table>

- E.g., *kernel* occurs about as often as *try* in total, but it occurs in fewer documents – it represents a more **specific** concept.
Inverse document frequency

• Relevant/specific words, \( w_i \), should give \textbf{smaller} values of \( df_i \).

• Therefore, the \textbf{inverse document frequency} is

\[
idf_i = \log \left( \frac{D}{df_i} \right)
\]

where \( D \) is the total number of documents and we scale with \( \log \), as before.

• This measure gives \textbf{full} weight to words that occur in 1 document, and \textbf{zero} weight to words that occur in all documents.
tf.idf

- We combine the term frequency and the inverse document frequency to give us a joint measure of relatedness between words and documents:

\[
tf.idf(w_i, d_j) = \begin{cases} 
(1 + \log(tf_{ij})) \log \frac{D}{df_i} & \text{if } tf_{ij} \geq 1 \\
0 & \text{if } tf_{ij} = 0
\end{cases}
\]
Aspects of tf.idf

• The \( tf. idf \) score has been criticised for being \textit{ad hoc}, but it has been shown to be robust and effective in a wide range of applications where a \textbf{rough} estimate of \textit{relatedness} is needed.

• However,
  • The effectiveness of \( tf. idf \) can vary depending on the \textbf{number of words} in a query.
  • \textbf{Ambiguous} words can cause spurious matches.
  • Vectors of \textit{terms} collected independently in this way can’t capture semantic \textbf{connections} between words.
    • We somehow need vectors of \textit{ideas/concepts}....
Latent semantic indexing

• **Co-occurrence**: *n.* when two or more terms occur in the same documents more often than by chance.
  • Note: this is *not* quite the same thing as collocations

• Consider the following:

<table>
<thead>
<tr>
<th></th>
<th>Term 1</th>
<th>Term 2</th>
<th>Term 3</th>
<th>Term 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>user</td>
<td>interface</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document 1</td>
<td>user</td>
<td>interface</td>
<td>HCI</td>
<td>interaction</td>
</tr>
<tr>
<td>Document 2</td>
<td></td>
<td></td>
<td>HCI</td>
<td>interaction</td>
</tr>
</tbody>
</table>

• Document 2 appears to be **related** to the query although it contains **none** of the query terms.
  • The query and document 2 are **semantically related**.
Latent semantic indexing

• **Latent semantic indexing** projects **queries** and **documents** into a space with **latent** (i.e., hidden) **semantic dimensions**.
  • Co-occurring terms are projected onto the **same semantic dimensions**. Non-co-occurring terms are not.
  • In semantic space, a query and a document can have high **cosine similarity** even if they do not share any terms.

• The latent semantic space has **fewer** dimensions than the original space (which had 1 dimension per term).
Latent semantic indexing

• There are many different mappings from high-dimensional spaces to low-dimensional spaces.

  • The ‘low dimensions’ are not usually selected from among the existing dimensions (in this case).

  • We must learn some function that projects our $N$ dimensions onto some new dimensions ...
Latent semantic indexing example

- This **original** space has 5 dimensions (one per term). The **reduced** space has two dimensions, perhaps vaguely referring to outer space and to vehicles, respectively.

<table>
<thead>
<tr>
<th></th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
<th>$d_5$</th>
<th>$d_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>cosmonaut</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>astronaut</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>moon</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>car</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>truck</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

What are some ways of projecting onto fewer dimensions?
1. Least-squares method

• Consider \( n \) points of functional data, \((x_1, y_1) \ldots (x_n, y_n)\). We want to fit the line \( f(x) = mx + b \) to these points.

• In a least-squares approach, the best fit is the one that minimizes the sum of the squares of the differences:

\[
\min_{m, b} SS(x, y; m, b) = \sum_{i=1}^{n} (y_i - (mx_i + b))^2
\]

• We find \( m \) and \( b \) that minimize this function by solving

\[
\frac{\delta SS(x, y; m, b)}{\delta m} = 0 \quad \text{and} \quad \frac{\delta SS(x, y; m, b)}{\delta b} = 0
\]
1. Least-squares method

- **Linear regression** reduces \((x_i, y_i)\) to just \(m\) and \(b\)
2. Principal components analysis (PCA)

- **PCA** is an eigendecomposition of the variance in the data.
  - It gives us a sequence of orthogonal vectors that represent the maximum amount of variance in the remaining data.

- Below, most of the variance is along $X$ – it captures the most amount of change in the data.
  - We can rotate the data so that it’s expressed in the new dimensions.
2. Principal components analysis (PCA)

- If we express the data *only* in terms of $X$,
  - it will be easier to learn a model.
    (we have fewer parameters)
  - we will lose some information.
    (what if the classes we want to distinguish are all differentiated in $Y$? (note: this should be rare))
3. Singular value decomposition

- **Singular value decomposition (SVD)** can be used both as:
  - a method of co-occurrence analysis between words that aids in similarity judgements, and
  - a method for dimensionality reduction.

- Just as **linear regression** projects 2-dimensional data onto a 1-dimensional line, SVD projects a $n$-dimensional matrix, $A$, onto a $k$-dimensional matrix, $\hat{A}$, where $n \gg k$.
  - The matrix $\hat{A}$ is produced such that some maximal amount of information in $A$ is retained.
3. Singular value decomposition

• The SVD projection is computed by decomposing the term-by-document matrix $A_{t \times d}$ into the product of three matrices:

$$T_{t \times n}, S_{n \times n}, \text{ and } D_{d \times n}$$

where $t$ is the number of words (terms),

$d$ is the number of documents, and $n = \min(t, d)$.

• Specifically,

$$A_{t \times d} = T_{t \times n} S_{n \times n} (D_{d \times n})^\top$$

Note: There are other formulations (dimensionalities), but the above is typical in IR.
3. Singular value decomposition

\[ M = U \cdot \Sigma \cdot V^* \]
SVD example

\[ A_{t\times d} = T_{t\times n} S_{n\times n} (D_{d\times n})^\top \]

\[
A = \begin{bmatrix}
\text{cosmonaut} & 1 & 0 & 1 & 0 & 0 & 0 \\
\text{astronaut} & 0 & 1 & 0 & 0 & 0 & 0 \\
\text{moon} & 1 & 1 & 0 & 0 & 0 & 0 \\
\text{car} & 1 & 0 & 0 & 1 & 1 & 0 \\
\text{truck} & 0 & 0 & 0 & 1 & 0 & 1 \\
\end{bmatrix}
\]

\[
S = \begin{bmatrix}
2.16 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.59 & 0 & 0 & 0 & 0 \\
0 & 0 & 1.28 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.39 & 0 \\
\end{bmatrix}
\]

\[
T = \begin{bmatrix}
\text{cosm.} & -0.44 & -0.30 & 0.57 & 0.58 & 0.25 \\
\text{astro.} & -0.13 & -0.33 & -0.59 & 0 & 0.73 \\
\text{moon} & -0.48 & -0.51 & -0.37 & 0 & 0.61 \\
\text{car} & -0.70 & 0.35 & 0.15 & -0.58 & 0.16 \\
\text{truck} & -0.26 & 0.65 & -0.41 & 0.58 & -0.09 \\
\end{bmatrix}
\]

\[
D^\top = \begin{bmatrix}
d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\
-0.75 & -0.28 & -0.20 & -0.45 & -0.33 & -0.12 \\
-0.29 & -0.53 & -0.19 & 0.63 & 0.22 & 0.41 \\
0.28 & -0.75 & 0.45 & -0.20 & 0.12 & -0.33 \\
0 & 0 & 0.58 & 0 & -0.58 & 0.58 \\
-0.53 & 0.29 & 0.63 & 0.19 & 0.41 & -0.22 \\
\end{bmatrix}
\]

- What do these matrices mean?
SVD example

\[
A = \begin{bmatrix}
\text{cosmonaut} & 1 & 0 & 1 & 0 & 0 & 0 \\
\text{astronaut} & 0 & 1 & 0 & 0 & 0 & 0 \\
\text{moon} & 1 & 1 & 0 & 0 & 0 & 0 \\
\text{car} & 1 & 0 & 0 & 1 & 1 & 0 \\
\text{truck} & 0 & 0 & 0 & 1 & 0 & 1 \\
\end{bmatrix}
\]

- \( A \) is the matrix of term frequencies, \( tf_{ij} \). E.g., \( \text{moon} \) occurs once in \( d_1 \) and once in \( d_2 \).
SVD example

- Matrices $T$ and $D$ represent terms and documents, respectively in this new space.
  - E.g., the first row of $T$ corresponds to the first row of $A$, and so on.

- $T$ and $D$ are orthonormal, so all columns are orthogonal to each other and $T^\top T = D^\top D = I$.

<table>
<thead>
<tr>
<th>term</th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
<th>$d_5$</th>
<th>$d_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>cosmo</td>
<td>-0.44</td>
<td>-0.30</td>
<td>0.57</td>
<td>0.58</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td>astro.</td>
<td>-0.13</td>
<td>-0.33</td>
<td>-0.59</td>
<td>0</td>
<td>0.73</td>
<td></td>
</tr>
<tr>
<td>moon</td>
<td>-0.48</td>
<td>-0.51</td>
<td>-0.37</td>
<td>0</td>
<td>-0.61</td>
<td></td>
</tr>
<tr>
<td>car</td>
<td>-0.70</td>
<td>0.35</td>
<td>0.15</td>
<td>-0.58</td>
<td>0.16</td>
<td></td>
</tr>
<tr>
<td>truck</td>
<td>-0.26</td>
<td>0.65</td>
<td>-0.41</td>
<td>0.58</td>
<td>-0.09</td>
<td></td>
</tr>
</tbody>
</table>

$T = \begin{pmatrix}
\text{cosm.} & -0.44 & -0.30 & 0.57 & 0.58 & 0.25 \\
\text{astro.} & -0.13 & -0.33 & -0.59 & 0 & 0.73 \\
\text{moon} & -0.48 & -0.51 & -0.37 & 0 & -0.61 \\
\text{car} & -0.70 & 0.35 & 0.15 & -0.58 & 0.16 \\
\text{truck} & -0.26 & 0.65 & -0.41 & 0.58 & -0.09 \\
\end{pmatrix}$

$D^\top = \begin{pmatrix}
d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\
-0.75 & -0.28 & -0.20 & -0.45 & -0.33 & -0.12 \\
-0.29 & -0.53 & -0.19 & 0.63 & 0.22 & 0.41 \\
0.28 & -0.75 & 0.45 & -0.20 & 0.12 & -0.33 \\
0 & 0 & 0.58 & 0 & -0.58 & 0.58 \\
-0.53 & 0.29 & 0.63 & 0.19 & 0.41 & -0.22 \\
\end{pmatrix}$
SVD example

- The matrix $S$ contains the **singular values** of $A$ in descending order.
  - The $i^{th}$ singular value hints at the amount of variation on the $i^{th}$ axis.

$$S = \begin{bmatrix}
2.16 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.59 & 0 & 0 & 0 & 0 \\
0 & 0 & 1.28 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.39 \\
\end{bmatrix}$$
**SVD example**

- By restricting $T$, $S$, and $D$ to their first $k < n$ columns, their product gives us $\hat{A}$, a ‘best least squares’ approximation of $A$.

\[
S = \begin{bmatrix}
2.16 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.59 & 0 & 0 & 0 & 0 \\
0 & 0 & 1.28 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.39 \\
\end{bmatrix}
\]

\[
D^T = \begin{bmatrix}
d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\
-0.75 & -0.28 & -0.20 & -0.45 & -0.33 & -0.12 \\
-0.29 & -0.53 & -0.19 & 0.63 & 0.22 & 0.41 \\
0.28 & -0.75 & 0.45 & -0.20 & 0.12 & -0.33 \\
0 & 0 & 0.58 & 0 & -0.58 & 0.58 \\
-0.53 & 0.29 & 0.63 & 0.19 & 0.41 & -0.22 \\
\end{bmatrix}
\]

\[
T = \begin{bmatrix}
\text{cosm.} & -0.44 & -0.30 & 0.57 & 0.58 & 0.25 \\
\text{astro.} & -0.13 & -0.33 & -0.59 & 0 & 0.73 \\
\text{moon} & -0.48 & -0.51 & -0.37 & 0 & -0.61 \\
\text{car} & -0.70 & 0.35 & 0.15 & -0.58 & 0.16 \\
\text{truck} & -0.26 & 0.65 & -0.41 & 0.58 & -0.09 \\
\end{bmatrix}
\]
SVD projection

• Recall that the purpose of doing SVD was to find co-occurrence patterns among terms and documents.

• When SVD finds the optimal projection to a lower-dimensional space, the result is that words that have similar co-occurrence patterns are projected onto the same dimensions.
  • e.g., car and truck could be projected on the same dimensions.

• Therefore, we can identify similarities between queries and documents even if they share no terms in common.

Note: PCA often uses SVD (e.g., it does in Matlab).
Similarities in lower dimensions

• Consider $B = S_{2 \times 2} \, D_{2 \times d}^\top$ from the previous example:

$B =$

\[
\begin{bmatrix}
  d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\
  -1.62 & -0.60 & -0.44 & -0.97 & -0.70 & -0.26 \\
  -0.46 & -0.84 & -0.30 & 1.00 & 0.35 & 0.65
\end{bmatrix}
\]

• If we normalize the columns of $B$ and multiply its transpose with it, we get:

\[
\begin{array}{cccccc}
  d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\
  d_1 & 1 & & & & \\
  d_2 & 0.78 & 1 & & & \\
  d_3 & 0.95 & 0.94 & 1 & & \\
  d_4 & 0.47 & -0.18 & -0.17 & 1 & \\
  d_5 & 0.74 & 0.16 & 0.49 & 0.94 & 1 \\
  d_6 & 0.10 & -0.54 & -0.22 & 0.93 & 0.75 & 1
\end{array}
\]
Similarities in lower dimensions

• This matrix gives us the correlation coefficients between documents in their latent (hidden) semantic space, regardless of their constituent terms.

\[
\begin{array}{cccccc}
 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\
\hline
 d_1 & 1 & & & & & \\
 d_2 & 0.78 & 1 & & & & \\
 d_3 & & 0.95 & 0.94 & 1 & & \\
 d_4 & 0.47 & -0.18 & 0.17 & 1 & & \\
 d_5 & 0.74 & 0.16 & 0.49 & 0.94 & 1 & \\
 d_6 & 0.10 & -0.54 & -0.22 & 0.93 & 0.75 & 1 \\
\end{array}
\]

• E.g., \(d_1\) and \(d_3\) are very similar, whereas \(d_1\) and \(d_6\) are not.
Latent semantic indexing in IR

• The process of projecting **different** (but related) terms to **common underlying dimensions** in the ‘**semantic**’ space can be thought of as ‘**soft clustering**’.
  • This is **desirable** – if a user poses a particular query, we want to identify all documents that might be of use, **regardless** of the specific terms used.
  • Occasionally this process results in some **spurious** matches, especially with polysemous (many meaning) words, but it remains a useful tool in practice.
Naïve Bayes

• Naïve Bayes can be used for *ad hoc* document retrieval.

• Imagine each of $D$ documents is a class with one training example – the document itself.

• Rank documents $d_j$ based on the *posterior probability* of a query $q$ “being generated” by those documents.

\[ P(q; d_j) \]

• This is called a “language modelling” approach.
Naïve Bayes – generative model

\[ P(q; d_1) = 0.3 \]

Ranked Retrievals:

\[
\begin{align*}
     d_1 & \quad 0.3 \\
     d_6 & \quad 0.28 \\
     d_5 & \quad 0.18 \\
     d_2 & \quad 0.1 \\
     d_3 & \quad 0.08 \\
     d_4 & \quad 0.06 \\
\end{align*}
\]
Smoothing

• Since each language model, $d_j$, will be very sparse, we need to smooth.

• **Linear interpolation:**

\[ P(q; d_j) \approx \lambda \hat{P}(q; d_j) + (1 - \lambda) \hat{P}(q) \]

where

\[ \hat{P}(q; d_j) \] is the probability of $q$ in the language model earned on $d_j$

\[ \hat{P}(q) \] is the probability of $q$ given the entire corpus.

• **Add-$\delta$:** same as in Assignment 2.
Experimental results

Linear interpolation performs almost as well as vector-space ranking (VSR).

Add-δ (called ‘Laplace’ here) – not so much.
Evaluation

• How can we decide which of two search engines is better at responding to a query?
  • **Precision** is the proportion of *returned documents* that are *relevant*.

  • **Recall** is the proportion of *relevant documents* that are *returned*.

  • **Rank** is also important – we want the correct documents to be near the top of the ordered list.
### Recall

- If there are 20 relevant documents in a collection, each of the following systems has a **recall** of \( \frac{5}{20} = 25\% \).

<table>
<thead>
<tr>
<th>Rank</th>
<th>Boogle</th>
<th>Ging</th>
<th>Whoopie</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>3</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>4</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>5</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>6</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>7</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>8</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>9</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>10</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
Precision is not enough

- Each of the following systems has a **precision** of 50%.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Boogle</th>
<th>Ging</th>
<th>Whoopie</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>3</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>4</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>5</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>6</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>7</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>8</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>9</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>10</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
Cutoff precision

- Measure precision down to some cutoff, e.g., first 5.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Boogle</th>
<th>Ging</th>
<th>Whoopie</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>3</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>4</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>5</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CP | 0% | 100% | 60%
**Uninterpolated Average Precision**

- Measure precision *at each relevant document returned.*

<table>
<thead>
<tr>
<th>Rank</th>
<th>Boogle</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>✓</td>
<td>1/6</td>
</tr>
<tr>
<td>7</td>
<td>✓</td>
<td>2/7</td>
</tr>
<tr>
<td>8</td>
<td>✓</td>
<td>3/8</td>
</tr>
<tr>
<td>9</td>
<td>✓</td>
<td>4/9</td>
</tr>
<tr>
<td>10</td>
<td>✓</td>
<td>5/10</td>
</tr>
</tbody>
</table>

**UAP**

\[
\frac{1/6 + 2/7 + 3/8 + 4/9 + 5/10}{5} = 0.3544
\]

Normalize at end by number of relevant documents.
## Uninterpolated Average Precision

<table>
<thead>
<tr>
<th>Rank</th>
<th>Boogle</th>
<th>Ging</th>
<th>Whoopie</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>6</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>8</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>9</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>10</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>UAP</strong></td>
<td>0.3544</td>
<td>1.0</td>
<td>0.6787</td>
</tr>
</tbody>
</table>
Interpolated average precision

• In *interpolated average precision (IAP)*, we compute *precision* relative to *recall*.

• Measure precision at fixed levels of recall
  • e.g., recall=0%, 10%, 20%,..., 100%  (the popular 11-point measure)

  • **For each** recall level \( r \), **find** the point in the ranked list where recall reaches \( r \), **then** compute precision \( p \).
  • **If** precision goes up to \( p_i \) as we move down the ranked list to recall level \( r_i \), **interpolate** by setting all recent smaller precisions to \( p_i \).
Example IAP

---

**Graph 1:**
- **X-axis:** Recall
- **Y-axis:** Precision
- The graph shows a curve that fluctuates between 0 and 1, indicating changes in precision with respect to recall.

**Graph 2:**
- **X-axis:** Recall
- **Y-axis:** Interpolated Precision
- This graph displays a nearly flat line with some fluctuations, indicating a stable interpolated precision across different recall values.
Interpolated average precision

- Measure precision at various levels of recall, then average.
- E.g., 11-point IAP with recall levels 0%, 10%, ..., 100%:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Boogle</th>
<th>Recall</th>
<th>UAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>✓</td>
<td>0%, 10%, 20%</td>
<td>1/6</td>
</tr>
<tr>
<td>7</td>
<td>✓</td>
<td>30%, 40%</td>
<td>2/7</td>
</tr>
<tr>
<td>8</td>
<td>✓</td>
<td>50%, 60%</td>
<td>3/8</td>
</tr>
<tr>
<td>9</td>
<td>✓</td>
<td>70%, 80%</td>
<td>4/9</td>
</tr>
<tr>
<td>10</td>
<td>✓</td>
<td>90%, 100%</td>
<td>5/10</td>
</tr>
<tr>
<td>11-pt IAP</td>
<td>SUM/11=0.3373</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In this example, there are only five relevant documents in total.

SUM/11=0.3373
## Interpolated average precision

<table>
<thead>
<tr>
<th>Rank</th>
<th>Boogle</th>
<th>Ging</th>
<th>Whoopie</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>6</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>8</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>9</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>10</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>11-pt IAP</td>
<td>0.337</td>
<td>1.0</td>
<td>0.708</td>
</tr>
</tbody>
</table>
Precision versus recall

• IAP is **more sensitive** to recall whereas UAP is more useful when your primary concern is relevance order.
  • Returning all documents available will give you 100% recall, but would be useless to the user.
  • Returning one document may give you high precision, but very low recall.

• The **F-score** combines precision and recall with parameter $\alpha$ that weighs the two:

$$F = \frac{1}{\frac{\alpha}{P} + \frac{1-\alpha}{R}}$$
Conclusion

• Reading: Manning & Schütze, chapter 15, especially sections 15.2 and 15.4.
  • Note: consider the errata for this textbook posted on the course webpage!

• Next: 11-1 Summarization  
  11-2 Miscellaneous machine learning and neural vector-space models  
  12-1 Review  
  18 April Exam